PRIMER 1

bez ugnjezdavanja

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class TestButton {
    public static void main(String[] args){
        ButtonFrame frame = new ButtonFrame();
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}

class ButtonFrame extends Jframe {
    public ButtonFrame()
    {
        setTitle("Test s bojom pozadine");
        setSize(300,200);
        Container cp = getContentPane();
        ButtonPanel panel = new ButtonPanel();
        cp.add(panel);
    }
}

class ButtonPanel extends JPanel implements ActionListener
{
    JButton yellow;
    JButton blue;
    JButton red;

    public ButtonPanel()
    {
        yellow = new JButton("Yellow");
        blue   = new JButton("Blue");
        red    = new JButton("Red");

        add(yellow);
        add(blue);
        add(red);

        yellow.addActionListener(this);
        blue.addActionListener(this);
        red.addActionListener(this);
    }

    public void actionPerformed(ActionEvent event) {
           Object obj = event.getSource();

           if(obj.equals(yellow))    setBackground(Color.YELLOW);
            else if(obj.equals(blue)) setBackground(Color.BLUE);
            else if(obj.equals(red))  setBackground(Color.RED);
    }
}
```

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

public class GridGUI {
     public static void main(String[] args){
         ButtonFrame frame = new ButtonFrame();
         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         frame.setVisible(true);
     }
}

class ButtonFrame extends JFrame
 {
     public ButtonFrame(){
         setTitle("Test s bojom pozadine");
         setSize(300,200);
         Container cp = getContentPane();
         ButtonPanel panel = new ButtonPanel();
         cp.add(panel);
     }
 }

class ButtonPanel extends JPanel {
     public ButtonPanel()
     {
         JButton yellow = new JButton("Yellow");
         JButton blue   = new JButton("Blue");
         JButton red    = new JButton("Red");

         add(yellow);
         add(blue);
         add(red);

         ColorAction yellowAction = new ColorAction(Color.YELLOW);
         ColorAction blueAction   = new ColorAction(Color.BLUE);
         ColorAction redAction    = new ColorAction(Color.RED);

         yellow.addActionListener(yellowAction);
         blue.addActionListener(blueAction);
         red.addActionListener(redAction);
     }

     // Privatna unutasnja klasa - ColorAction() konstruktor ne mora
     // dobiti referencu na ButtonPanel koja bi joj trebala da dohvati
     // ButtonPanel.setBackground(backgroundColor)

     private class ColorAction implements ActionListener
     {   private Color backgroundColor;
         public ColorAction(Color c) { backgroundColor=c; }

         public void actionPerformed(ActionEvent e) {
             // metoda iz JComponent klase
             setBackground(backgroundColor);
         }
     }
 }
```

ili

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;


public class GridGUI {
     public static void main(String[] args){
         ButtonFrame frame = new ButtonFrame();
         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
         frame.setVisible(true);
     }
 }

class ButtonFrame extends Jframe {
     public ButtonFrame(){
         setTitle("Test s bojom pozadine");
         setSize(300,200);
         Container cp = getContentPane();
         ButtonPanel panel = new ButtonPanel();
         cp.add(panel);
     }
 }

 class ButtonPanel extends JPanel {
     public ButtonPanel() {
        makeButton("Yellow", Color.YELLOW);
         makeButton("Blue", Color.BLUE);
         makeButton("Red", Color.RED);
     }
     void makeButton(String labela, final Color bojaPozadine)
     {
         JButton gumb = new JButton(labela);
         add(gumb);
         gumb.addActionListener(new
                 ActionListener() {
                      public void actionPerformed(ActionEvent e) {
                      setBackground(bojaPozadine);
                 }
                 });
     }

 }
```

PRIMER 2

```
import java.awt.event.*;
import javax.swing.*;

public class TestWindowListener {
    public static void main(String[] args){
        SmartFrame frame = new SmartFrame();
        frame.setDefaultCloseOperation(JFrame.DO_NOTHING_ON_CLOSE );
        frame.setVisible(true);
    }
}

class SmartFrame extends Jframe {
    public SmartFrame()
    {
        setTitle("Prazan prozor !?");
        setSize(300,200);
        WindowListener wl = new Terminator();
        addWindowListener(wl);
    }
}

class Terminator extends WindowAdapter {
    public void windowClosing(WindowEvent e){
     int i=JOptionPane.showConfirmDialog(null, "Zatvoriti ili ne zatvoriti???",
                                          "Exit", JOptionPane.YES_NO_OPTION,
                                           JOptionPane.WARNING_MESSAGE);
        if(i == JOptionPane.OK_OPTION)
            System.exit(0);
    }
}
```

PRIMER 3

```
	import java.awt.event.*;
	import javax.swing.*;
	public class TestWindowListener {
		public static void main(String[] args) {
		MyFrame frame = new MyFrame();
		frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE );
		frame.setVisible(true);
		}
	}
// Svaka komponenta moze biti KeyListener pa stoga
// necemo kreirati panel vec ce frame slusati tastaturu.
	class MyFrame extends JFrame implements KeyListener {
		public MyFrame() {
			setTitle("Skupljam ulaz s tastature");
			setSize(300,200);
			addKeyListener(this);
		}
		public void keyTyped(KeyEvent e) {
			char c = e.getKeyChar();
			System.out.println("keyTyped : znak = "+c);
		}
		public void keyReleased(KeyEvent e) {
			int kod = e.getKeyCode();
			System.out.println("keyReleased: kod = "+kod);
		}
		public void keyPressed(KeyEvent e) {
			int kod = e.getKeyCode();
			System.out.println("keyPressed : kod = "+kod);
		if(kod == KeyEvent.VK_SHIFT)
			System.out.println("keyPressed : SHIFT pritisnut.");
		if(kod == KeyEvent.VK_C && e.isShiftDown() && e.isControlDown())
			System.out.println("keyPressed : SHIFT_CTRL_C pritisnut");
		}
	}
```

PRIMER 4

```
import java.awt.*;
import java.awt.event.*;
import java.awt.geom.*;
import javax.swing.*;
import java.util.*;
public class TestMouseListener {
	public static void main(String[] args) {
		MouseFrame mf = new MouseFrame();
		mf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		mf.setVisible(true);
	}
}
class MouseFrame extends JFrame {
	public MouseFrame() {
		setTitle("MouseFrame");
		setSize(300,200);
		MousePanel mp = new MousePanel();
		Container contentPane = getContentPane();
		contentPane.add(mp);
	}
}
class MousePanel extends JPanel {
	private static final int DUZINA = 10;
	private ArrayList kvadrati; // lista kvadrata
	private Rectangle2D trenutni; // aktuelni kvadrat
	public MousePanel() {
		kvadrati = new ArrayList();
		trenutni = null;
		addMouseListener(new MouseHandler());
		addMouseMotionListener(new MouseMotionHandler());
	}
//	Iscrtavanje panela
	public void paintComponent(Graphics g) {
		super.paintComponent(g);
		Graphics2D g2 = (Graphics2D) g;
		for(int i=0; i<kvadrati.size(); ++i)
			g2.draw((Rectangle2D) kvadrati.get(i));
	}
//	Rutine za manipulaciju s listom kvadrata: add, find, remove
//	Metode se jednostavno implemetiraju pomocu metoda klase ArrayList

//	Dodaj novi kvadrat s centrom u tacki p.
	public void add(Point2D p) {
		double x = p.getX();
		double y = p.getY();
		trenutni = new Rectangle2D.Double(x-DUZINA/2, y-DUZINA/2,
				DUZINA, DUZINA);
//		Koristimo metodu add iz ArrayList
		kvadrati.add(trenutni);
		repaint();
	}
//	Pronadji element u listi koji sadrzi tacku p. Vrati null ako takvog nema.
	public Rectangle2D find(Point2D p) {
//		Sav posao odradjuje metod contains iz Rectangle2D koja ispituje je li
//		tacka unutar pravokutnika.
		for(int i=0; i<kvadrati.size(); ++i){
//			moramo castati -- u Java 5.0 moze elegantnije ako koristimo
//			ArrayList<Rectangle2D>
```

```
                        Rectangle2D rec=(Rectangle2D) kvadrati.get(i);
                        if(rec.contains(p)) return (Rectangle2D) kvadrati.get(i);
                }
                return null;
        }
//      Odstrani element iz liste
        public void remove(Rectangle2D r) {
                if(r == null) return;
                if(r == trenutni) trenutni = null;
                kvadrati.remove(r);
                repaint();
        }
//      Rutine za procesiranje dogadjaja. Smjestene su u dvije unutrasnje klase.
//      Privatna unutrašnja klasa
        private class MouseHandler extends MouseAdapter {
//              Cim pritisnemo tipku misa kreiramo novi kvadrat
                public void mousePressed(MouseEvent e) {
//                      Da li se pritisak dogodio unutar nekog pravokutnika?
                        trenutni = find(e.getPoint());
                        if(trenutni == null) // nije
                                add(e.getPoint()); // dodaj novi pravokutnik
                }
//              Ako kliknemo dvaput u kvadratu brisemo ga
                public void mouseClicked(MouseEvent e) {
//                      Da li se pritisak dogodio unutar nekog pravokutnika?
                        trenutni = find(e.getPoint());
                        if(trenutni != null && e.getClickCount() >=2 ) // da, bar dva puta
                                remove(trenutni); // brisi pravokutnik
                }
        }
//      Privatna unutrasnja klasa
        private class MouseMotionHandler implements MouseMotionListener {
                public void mouseMoved(MouseEvent e) {
//                      Ako se kretanje desava unutar kvadrata promijeni kursor
                        if(find(e.getPoint()) == null)
                                setCursor(Cursor.getDefaultCursor());
                        else
                                setCursor(Cursor.getPredefinedCursor(Cursor.CROSSHAIR_CURSOR));
                }
                public void mouseDragged(MouseEvent e) {
//                      Cim se stisne tipka unutar nekog pravougaonika
//                      bit ce postavljen trenutni
                        if(trenutni != null)
                        {
                                int x = e.getX();
                                int y = e.getY();
//                              Vucemo kvadrat
                                trenutni.setFrame(x-DUZINA/2, y-DUZINA/2, DUZINA, DUZINA);
                                repaint();
                        }
                }
        }
}
```